Instructions:

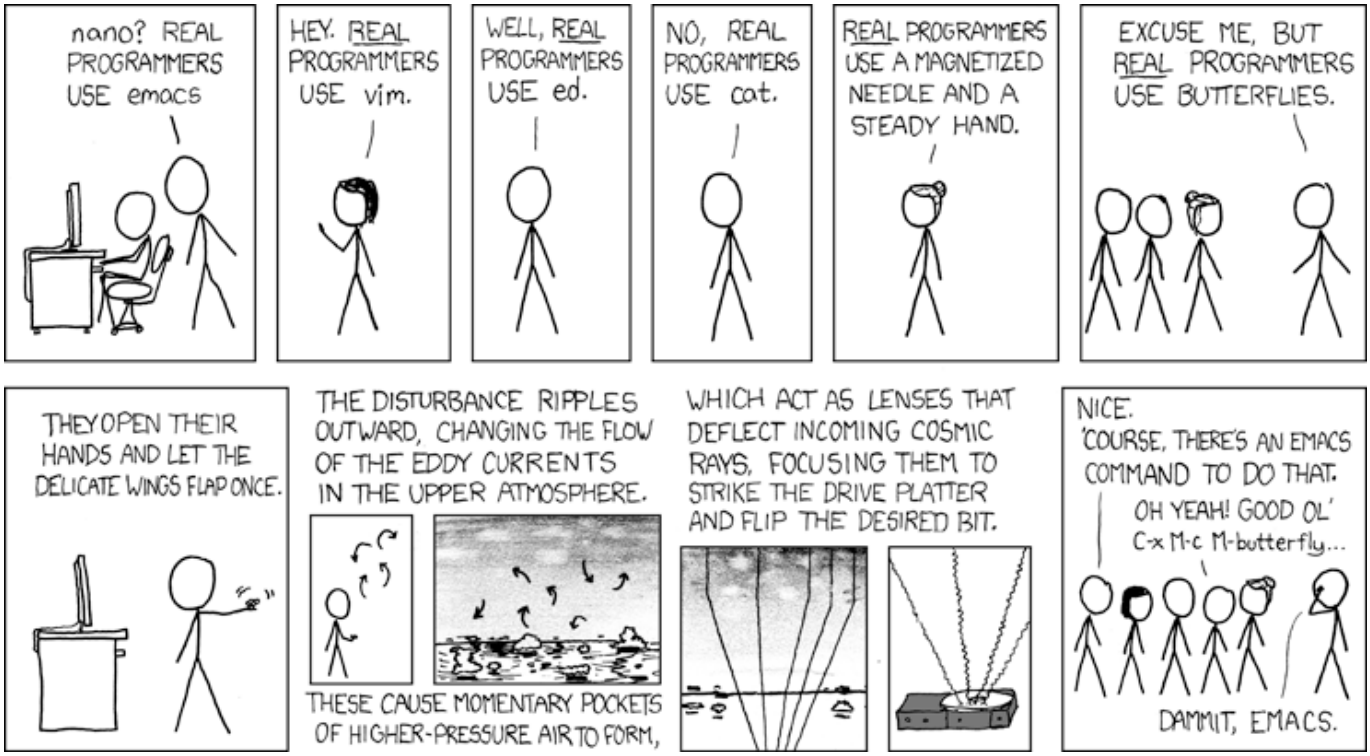
- Print your name in the space provided below.
- This examination is closed book and closed notes. No calculators or other computing devices may be used. The use of any such device will be interpreted as an indication that you are finished with the test and your test form will be collected immediately.
- Answer each question in the space provided. If you need to continue an answer onto the back of a page, clearly indicate that and label the continuation with the question number.
- If you want partial credit, justify your answers, even when justification is not explicitly required.
- There are 7 questions, some with multiple parts, priced as marked. The maximum score is 100.
- When you have completed the test, sign the pledge at the bottom of this page and turn in the test.
- Note that either failing to return this test, or discussing its content with a student who has not taken it is a violation of the Honor Code.

Do not start the test until instructed to do so!

Name _____
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_____
signed


nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

xkcd.com

1)

a) [8 points] What would be the output from executing the Python code below? Be specific, vague answers will receive no credit. Either write out the entire output (safest), or provide enough information that I can tell what's on each line.

```
for x,c in enumerate(("HelloWorld"*3)[10:22:3]):
      print(x, c)
```

```
0 H
1 l
2 o
3 d
```

b) [6 points] What would be the output from executing the Python code below? Be specific, vague answers will receive no credit. Either write out the entire output (safest), or provide enough information that I can tell what's on each line.

```
# A space character being used with join
s =  " ".join([“CS1064", "Fall",  "2014”])
print(s[0:3] + s[5:6])
print(s[0:len(s):2])
```

```
CS14
C16 al21
```

2)

a) [8 points] What would be the output from executing the Python code below? Be specific, vague answers will receive no credit. Either write out the entire output (safest), or provide enough information that I can tell what's on each line.

```
test_list = [3, 7, 6, 2]

print(test_list[0:3])
print(test_list[::-1])
print(test_list.sort())
print(test_list + [5])
```

```
[3, 7, 6]
[2, 6, 7, 3]
None
[2, 3, 6, 7, 5]
```

b) [8 points] What would be the output from executing the Python code below? Be specific, vague answers will receive no credit. Either write out the entire output (safest), or provide enough information that I can tell what's on each line.

```
test_list = [1, 2, 3, 4]

for x in range(4, 15, 3):
      print(test_list[x % len(test_list)])
```

```
1
4
3
2
```

3) **[10 points]** Create a Python program that will **remove** a single character from a string at a user specified position or index (i.e. the position comes from `input`) and then print the result.

So given the string:

```
"CS1604 Fall 2014"
```

If the user wants to remove position 3 in the string, your code should print:

```
"CS104 Fall 2014"
```

```
# Program starts

# Get the position, done for you
pos = int(input("Give me a position: "))


# the string value comes from somewhere (a file, user input, etc)
# the actual values shouldn't be important to your code/algorithm
remove_pos = "..."

# Your code using remove_pos and pos goes here


remove_pos = remove_pos[:pos] + remove_pos[pos+1:]
print(remove_pos)
```

4) **[15 points]** Create a Python program that will "double" the white space inside of a string. So every for position that there is white space, in the result there would be 2 white space characters. This should work with any white space characters not just spaces. Hint: You may want to `import string`.

So given the string:

```
"CS1604_Fall_2014"
```

Your code should print:

```
"CS1604__Fall__2014"
```

The underscores in this example are used only **for clarity**; you want to find **only** the whitespace (not underscores) and it double it.

```
# Program starts
import string

# the string value comes from somewhere (a file, user input, etc)
# the actual values shouldn't be important to your code/algorithm
double_ws = "..."

# your code using double_ws goes here
```

```
all_chars =""

for c in double_ws:
      if c in string.whitespace:
            all_chars += 2*c
      else:
            all_chars += c

print(all_chars)
```

5) **[15 points]** Create a Python program that will flip the case of each letter in a string. Numbers, symbols, and white space should be ignored, and otherwise print normally.

So given the string:

```
"CS1604 Fall 2014"
```

Your code should print:

```
"cs1604 fALL 2014"
```

```
# Program starts
import.string

# the string value comes from somewhere (a file, user input, etc)
# the actual values shouldn't be important to your code/algorithm
flip_case = "..."

# Your code using flip_case goes here


all_chars =""

for c in flip_case:
      if c.isupper():
            all_chars += c.lower()

      elif c.islower():
            all_chars += c.upper()

      else:
            all_chars += c

print(all_chars)
```

6) **[15 points]** Write a Python program that finds and then prints the **index** of the **minimum** integer value in the list. You may not use any built-in list method or function (with the exception `len()`), i.e. you must iterate through the list yourself and find the position of the minimum value. If the list is empty print 0, if there are two minimum values print the index of earliest one.

So if the list contains:

```
test_list = []               # your code would print 0
test_list = [8, 5, 3, 10]    # your code would print 2
test_list = [8, 7, 10, 7]    # your code would print 1

# Program begins

# test_list comes from somewhere (a file, user input, etc)
# the actual values shouldn't be important to your code/algorithm
test_list = [?, ?, ?, ...]

# Your code using test_list goes here


min_idx = 0

for x in range(len(test_list)):
      if test_list[x] < test_list[min_idx]:
            min_idx = x


print(min_idx)
```

7) **[15 points]** Write a Python program that opens and processes a file. The file will contain an arbitrary number of lines, where each line will either contain 1 or more integer values or will be blank. The file below has 3 lines containing integers and one blank line.

```
1 2 3 4

1 1 2
1 3 3 5
```

If the line contains an even number of integers, your program should sum the integers on **that line** and print the result. If the line contains an odd number of integers, your program should average the integers on **that line** and print the result. A blank line in the file should print "Nothing on this line".

Given the file above your program should print:

```
10
Nothing on this line.
1.3333333333333333
12
```

```
# Program starts
input_file = open("input.txt", "r")

# Your code goes here, should use input_file


for line in input_file:

      sum = 0
      numbers = line.split()

      for num in numbers:
            sum += int(num)

      if numbers == []:
            print("Nothing on this line.")

      elif len(numbers) % 2 == 0:
            print(sum)

      elif len(numbers) % 2 == 1:
            print(sum/len(numbers)
```